1

- *Merhaba Qt*
- *Bağlantıların yapılışı*
- *Referans dokümantasyonun kullanımı*

Başlarken

Bu bölümde temel C++ ile Qt tarfından tedarik edilen fonksüyonları bir araya getirerek nasıl grafik kullanıcı arabirimi (GUI) yazılabileceğini bir kaç basit misal ile göstereceğiz.
Bu bölümde yine Qt için çok önem arzeden ve iki temel taşı olan *sinyaller ve dilimler* ile *dizim* den bahsedilecektir. İkinci bölümde detaylara ineceğiz ve üçüncü bölümden itibaren artık gerçekçi proğramlar yazıyor olacağız.

Merhaba Qt

İşte çok basit bir Qt proğramı:

```
 1 #include <qapplication.h>
 2 #include <qlabel.h>
 3 int main(int argc, char *argv[])
 4 {
 5    QApplication app(argc, argv);
 6    QLabel *label = new QLabel("Merhaba Qt!", 0);
 7    app.setMainWidget(label);
 8    label->show();
 9    return app.exec();
10 }
```

Önce biz bu proğramı satır satır inceleyeceğiz ve daha sonrada nasıl *derlenebileceğını* göstereceğiz.

Satır 1 ve 2 de QApplıcatıon ve QLabel sınıflarının tnımları programa dahil edilmektedır. Satır 5 de bir QApplicatıon nesnesi oluşturulmaktadır ki bu nesne proğramın temelini teşkil eder. QApplicatıon sınıfının yapıcısı *argc* ve *argv* bağımsız değişkenlerini gerektiriyor, çünkü Qt nin kendine has birkaç komut satırı seçenekleri var.

Satır 6 da bir QLabel *widget* (vicıt diye telaffuz edilir) olusturuyor ki o “Merhaba Qt” mesajını ekrana yansıtıyor. Qt ıstılahatında, *widget* kullanıcı arabiriminde grafıksel bir birimdir. Düğmeler, menüler, kaydrma çubuklar ve çerçeveler den her biri birer wdgetlardr. Bir wıdget bir başka wıdget ihtiva edebilir; mesela bir yazılım genellikle QMenuBar, QToolBar

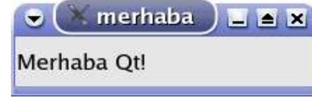
ve QStatusBar ihtiva etmekle beraber daha başka wıdgetlarıda içerebilir. QLabel yapıcısının 0 bağımsız değişkeni bu widget ın baska bir wıdget içerisinde yer almayıp başlı başına bır pencere teşkil ettiği anlamına gelir.

Satır 7 label'ı bu proğramın ana widgetı yapar. Ne zamanki kullanıcı ana wıdgeti kapatırsa (pencerenin baslığındaki X'e tıklamak suretiyle mesela) o zaman proğram sona erer. Ana wıdgeti olmayan bir program kullanıcı her ne kadar pencereyi kapatsada arka planda calışmaya devam eder.

Satır 8 label'in ekranda g.özükmesini sağlar. Widgetlar ilk yapıldıklarında (satır 6) ekranda gözükmezler taki tüm degişiklikler arka planda yapılsın ve ekranda titremelere neden olmasın.

Satır 9 proğramın kumandasını Qt'ye verir. Bu aşamada program artık hazır bekleme vaziyeti alır ve tuşa basılması veya fare tıklaması gibi komutları bekler.

Kullanıcının eylemine olaylar (yada mesajlar) meydana getirir ve proğram bunlara bir veya daha fazla fonksıyonu koşturarak karşılık verir. Çalışmaları itibariyle GUI proğramları alışılmış toplu işlem proğramlarından farklıdırlar. Toplu işlem proğramları genelde girdiyi alır, sonucu üretir ve kullanıcı temasına gerek kalmadan sona erer.



Şekil 1.1: Merhaba (Red Hat Linux)

Şimdi artık proğramı sizin bigisayarınızda denemenin zamanı. Önce Ek A da açıklandığı gibi Qt 3.2 (yada daha yeni bir versiyonu) bilgisayarınıza kurmanız gerekli. Şu andan itibaren biz sizin bilgisayarınızda Qt 3.2 nin kurulmuş olduğunu ve bin dizininin PATH değişkeninde yer aldığını varsayacağız. (Widonwsda bu kurucu tarafından otomatık olarak yapıldıgından sizin herhangi bir şey yapmanıza gerek yok.)

Merhaba proğramının kaynak kodunu *merhaba* isminde bir dizinin içinde *merhaba.cpp* ismi ile kaydeiniz. Peoğramın kodunu bilgisayarda yazabileceğiniz gibi kitapla birlikte gelen CD den de kopyalayabilirsiniz (*\misaller\bab01\merhaba\merhaba.cpp*.)

Her hangi bir komut isteminden merhaba isimli dizime geçiniz ve

```
qmake -project
```

yazınız ve böylece işletim sisteminden bağımsız proje dosyasını (merhaba.pro) oluiturmuş olacaksınız. Daha sonra

```
qmake merhaba.pro
```

yazmak suretiyle kullandığınız işletim sistemine has Makefile oluşturunuz. Eğer Makefile

oluşturulurken bir hata zuhur etmedıyse, *make* yazmak suretiyle proğramı derleyebilirsiniz. Yeni oluşturduğunuz bu proğram Windows altında merhaba ve unix altında ./merhaba yazılarak koşturulabilir. Şayet Microsoft Visual C++ kullanıyorsanız, *make* yerine *nmake* kullanmalısınız. Windows altında

```
qmake -tp vc merhaba.pro
```

yazmak suretiyle Visual Studio proje dosyası oluşturup derleme işlemini Visual Studio altındada yapabilirsiniz.



Şekil 1.2. Basit HTML formatı içeren bir label (etiket)

Gelin biraz eğlenelim şimdi: Etiketi, basit bir takım HTML formatları kullanmak suretiyle, renklendireceğiz (Şekil 1.2). Bunu

```
QLabel *label = new QLabel("Hello Qt!", 0);
```

satırının yerine

```
QLabel *label = new QLabel("<h2><i>Hello</i> "
                     "<font color=red>Qt!</font></h2>", 0);
```

satırını kullanmak suretiyle gerçekleştirebiliriz.

Bağlantıların yapılışı

Bu misal kullanıcı eylemlerine nasıl karşılık verilebileceğini göstermektedir. Bu örnek tıklandığında proğramı durduracak bir düğmeden ibarettir. Kaynak kodu Merhaba proğramına çok benzemekle birlikte tek fark QLabel yerıne QPushButton ana wıdget olarak kullanılması ve bir kullanıcı eylemının (düğmeye tıklama) bir parça koda bağlanmasıdır.

Kaynak kodu CD de *\misaller\bab01\cik\cik.cpp* dosyasında bulabilirsiniz.



Şekil 1.3. Çık proğramı

```
#include <qapplication.h>
#include <qpushbutton.h>
int main(int argc, char *argv[])
{
```

```
    QApplication app(argc, argv);
    QPushButton *button = new
          QpushButton(QObject::trUtf8("Çık"), 0);
    QObject::connect(button, SIGNAL(clicked()),
                                  &app, SLOT(quit()));
    app.setMainWidget(button);
    button->show();
    return app.exec();
}
```

Qt nin widgetları kullanıcı faliyetlerini ve durum değişikliklerini bildirmek maksadıyla *sinyaller*[*] neşrederler. Mesela QPushButton düymeye basıldığında clıcked() diye bir sinyal yayar. Sinyal bir fonksiyona bağlanabilir. Bir sinyalin bağlanabildiği fonksiyona *dilim* adı verilir ve ne zamanki o sinyal neğredilirse bağlı bulunduğu dilim otomatik olarak icra edilir. Bizim misalimizde düğmenin cicked() sşnyalini Qapplicatioon nesnesinin quit() dilimine bağladık. . SIGNAL() ve SLOT() makroları bağlantıların bölünmez bir parçasıdır ve bir sonraki bölümde detaylı olarak ele alınacaklardır.

Şimdi proğramı derleyeceğiz. Biz sizin cik isminde bir dizi oluşturduğunuzu ve bı dizinin cik.cpp isminde proğramı içerdiğini varsayacağız. Önce qmake koşturarak proje dosyasını oluşturunuz, sonra yine qmake kullnarak Makefile yapınız. Bütün bunları cik dizisinin içindeyken yapmalısınız:

```
qmake -project

qmake cik.pro
```

Şimdi proğramı derle ve koştur. Eğer çık düğmesine tıklayacak yada ara tuşuna basacak olursanız programın sona erceğini göreceksiniz.

Bir sonrakı misal *sinyal* ve *dilim* mekanizmasını kullanarak nasıl iki wıdgetin birbiriyle beraber hareket etmelerinin sağlanabileceğini göstermektedir. Proğram kullanıcıya yaşını sormaktadır ve o da ya fırıldak kutusu yada kaydırıcı vasıtasıyla girebilir.



Şekil 1.4. Yaş proğramı

Bu proğram bir QSpinBox, bir QHBox (yatay dizim kutusu) ve bir Qslider dan müteşekkildir. QHBox proğramın ana widgetıdır. QSpinBox ve QSlider QHBox ın içerisinde görüntülenmektedirler bundan dolayı onlar QHBox ın çocuklarıdırlar.

[*] Qt nin sinyalleri ile Unix sinyalleri arasında her hangi bir ilişkı olmamakla beraber, biz bu kıtapda sadece Qt sinyalleri üzerinde duracağız.



Şekil 1.5. Yaş proğramının widgetleri

```
 1 #include <qapplication.h>
 2 #include <qhbox.h>
 3 #include <qslider.h>
 4 #include <qspinbox.h>
 5 int main(int argc, char *argv[])
 6 {
 7    QApplication app(argc, argv);
 8    QHBox *hbox = new QHBox(0);
 9    hbox->setCaption(QObject::trUtf8("Yaşnızı giriniz"));
10    hbox->setMargin(6);
11    hbox->setSpacing(6);
12    QSpinBox *spinBox = new QSpinBox(hbox);
13    QSlider *slider = new QSlider(Qt::Horizontal, hbox);
14    spinBox->setRange(0, 130);
15    slider->setRange(0, 130);
16    QObject::connect(spinBox, SIGNAL(valueChanged(int)),
17    slider, SLOT(setValue(int)));
18    QObject::connect(slider, SIGNAL(valueChanged(int)),
19    spinBox, SLOT(setValue(int)));
20    spinBox->setValue(35);
21    app.setMainWidget(hbox);
22    hbox->show();
23    return app.exec();
24 }
```

QHBox, 8 ile 11 inci satırlar arasında oluşturulduktan sonra, setCaprion() fonksiyonunu kullanarak baslık çubuğunun metni değiştiriliyor. Daha sonra çocukların aralarında ve etraflarında 6 piksellik bir boşluk bırakıyoruz.

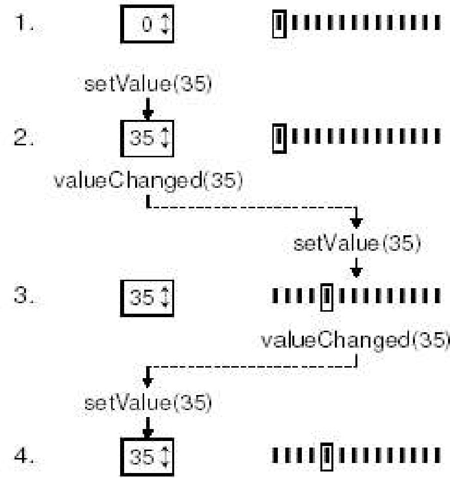
Satır 12 ve 13 de bir QSpinBox ve birde QSlider oluşturulmaktadır kı her ikisininde ebeveyni QHBox dır.

Her ne kadar biz her hangi bir widgetin yerini ve boyutlarını bizzat vermedıysekte, QSpinBox ve Qslider, QHBox ın içinde gayet makul bir şekilde ve büyüklükte yerleştirilmiş olduklarını görüyoruz. Bunun sevebı QHBox ın çocuklarının her biri için gerekli olan ebatı ve müsaıt makamı otomatğk olarak vermesidir. Qt, QHBox gibi daha bir çok sınıflar tedarik etmektedirkı buda proğramcıyı doğrudan wıdgetlerin ebat ve yerlerini belirleme yükünden kurtarır.

Satır 14 ile 15 de geçerli fırıldak kutusu ve kaydırıcı limitleri verilmektedir. Onaltıncı ve

ondokuzuncu satırlardakı connect() fonksiyon çağrıları, fırıldak kutusu ve kaydırıcının birbirleri ile irtibat halinde olmalarını sağlarlar ve birinin degeri degiştirildiğinde diğerinin değeride otomatik olarak değişir. Bir widgetin değeri depişdiğinde, o widget valueChanged (int) sinyalini yayınlar ve bu diğer widgetin setValue(int) diliminin çağrılmasına neden olur ki bu fonksiyon sözkonusu widgetin değerini güncelleştirir.

Satır 20 de fırıldak kutusunun değeri 35 yapılır. Dikkat edilirse biz kaydırıcının değerini girmedik, bunun sebebi ise 20. satırda olup bitenler fırıldak kutusunun valueChanged(int) sinyalinin 35 değişkeniyle yayınlamasına sbep olacak ve buda QSlider widgetının setValue (int) dilimine değişken olarak gönderilecek neticede kaydırıcının değeri 35 olarak ayarlanmış olacak. Kaydırıcının değeri değişince oda valueChanged(int) sinyalini yayınlıyor buda fırıldak kutusunun setValue(int) diliminin çağrılmasına neden oluyor. Ancak fırıldak kutusu bu noktada 35 degerini tuttuğu için o artık değerinin değiştiğine dair herhangi bir yayın yapmaz. Böylece kısır döngü engellenmiş olur. Şekil 1.6 bu meseleyi özetlemektedir.



Şekil 1.6 Bir değeri değiştirmeye kalkınca her iki değer birden değişiyor

Satır 22 QHBox ve iki çocuğunu görüntüler. Qt nin kullanıcı arabirimi oluşturulmasındaki yaklaşımı kolay anlaşılabilir ve gayet esnektir. Qt proğramcıları genelde gerekli widgetları oluşturduktan sonra onların özellilikleri istedikleri şekilde değiştirirler. The most common pattern that Qt programmers use is to instantiate the required widgetsand then set their propertiesas necessary. Proğramcılar widgetları dizimlerin içerisine yerleştirirler ve dizimler otomatik olarak widgetların ebat ve ekrandaki yerlerini ayarlarlar. Kullanıcı arabirimlerinin davranışları Qt nin *sinyaller* ve *dilimler* mekanızması sayesinde widgetlarin birbirlerine rapdtedilmeleri ile sağlanır.

Referans dokümantasyonun kullanımı

Qt nin referans dokümantasyonu*,bütün fonksiyon ve sıfları ihtiva etmesı sebebiyle, Qt proğramcıları için çok büyük bir önem arzeder. Qt 3.2, örneğin, 400 den faxla sınıf ve 6000 den fazla fonksiyonu ihtiva eder. Bu kitaptaki misallerde çok sayıda Qt sınıf ve fonksiyonları

* Kanatimizce Qt referance dokümantasyonunun, malesef, sadece İngilizce versiyonu mevcuttur. Bu kitap bildiğimiz kadarıyla lisanımızla yayınlanan en kapsamlı Qt referansıdır bunun böyle ittihaz edilmesini temenni ederiz.

kullanılmaktadır ancak bunlardan bütün detayları ile bahsetmek yada bütün sınıfları ve fonksiyonları misallerle anlatmak malesef bu kitabın sahası dahilinde değildir. Bundan dolayı referans dokümantasyonunu behemehal kullanma durumunda olacaksınız.

Qt nin dokümantasyonu *doc\html* dizini altında mevcut olup har hangi bir web gözatıcısı kullanılarak erişilebilir. Qt Assistant isminde Qt ile gelen yaedım gözatıcısı Qt nin yardım dosyalarını okumak için gayet elverişlidir. Windiws da Başla menüsünün altında, Qt menüsünden Qt Assıstant başlatılabilir. Unix altında ise komut isteminde assistant yazılarak yardım dosyalarına ulaşabilirsiniz, ve Mac OS X Bulucusu içinde assistant a iki defa tıklayınız.

Ana sahifedeki *API Reference* başlığı altında mevcut olan bağlantılar bir çok bakımdan Qt sınıflarını incelemenize ımkan sağlarlar. *All Classes* başlığı altında Qt de mevcut olan butün sınfları bulabilirsiniz. *Main Classes* sayfası sadece en öok kullanılan ana sınflar hakkında bilgi içermektedir. Alıştırma olması bakımından şu ana kadar misallerde kullandığımıx sınıflar hakkında referans dokümantasyonunu kullanarak bilgi edinmeye çalış. Şuna dıkkat edilmelidirki, kalıtsal (inherited) fonksiyonşar ana sınıf altında tasvir edilmektedirler; mesela, QPushButton sınıfının kendine has show() diye bir fonksüyonmu olmayıp aksine o ona QWideget sınıfından miras kalmıştır. Şekil 1.10 da şu ana kadar karşılaştığımız sınıfların birbirlerine nasıl mutaallik olduklarını müşahade edebilirsiniz.

Qt nin hali hazırdaki versiyonu ve bir kısım eski versiyonlarının referans *dokümantasyonlarına* internet de http://doc.trolltech.com/ adresinden ulaşabilirsiniz. Yine bu sayfada Qt tarafından her üç ayada bir çıkarılan *Qt Quarterly* ile *Qt programmers newsletter* adındaki cerideyi burada bulabilirsiniz.

Widget Tipleri

Şu ana kadar gördüğümüz ekran resimleri Redhat Linux altında alınmıştır, ancak Qt proğramlarına desteklenen işletim sistemlerinden istediğiniz herhangi birisinin görüntüsünü verebilirsiniz. Qt nin bu olanağı sağlayabilmesindeki tılsım, onun her bir platformun kendıne has araç kutularını kullanmak yerine (Windows XP ve Mac hariç) onlari taklit etmesidir.



Windows



Motif



MotifPlus



CDE



Platinum



SGI

Şekil 1.7: Her işletim sisteminde kullanılabşlecek tipler.

Qt prğramlarının görünümü *-style* komut satırı seçeneğini kullanmak suretiyle değiştirilebilir. Mesela, daha önce gördüğümüz yaş proğramına Unix işletim sistemi altında Platinum görünmü komut satırında aşağıdakı komutu kullanarak verilebilir.

```
./yas -style=Platinum
```



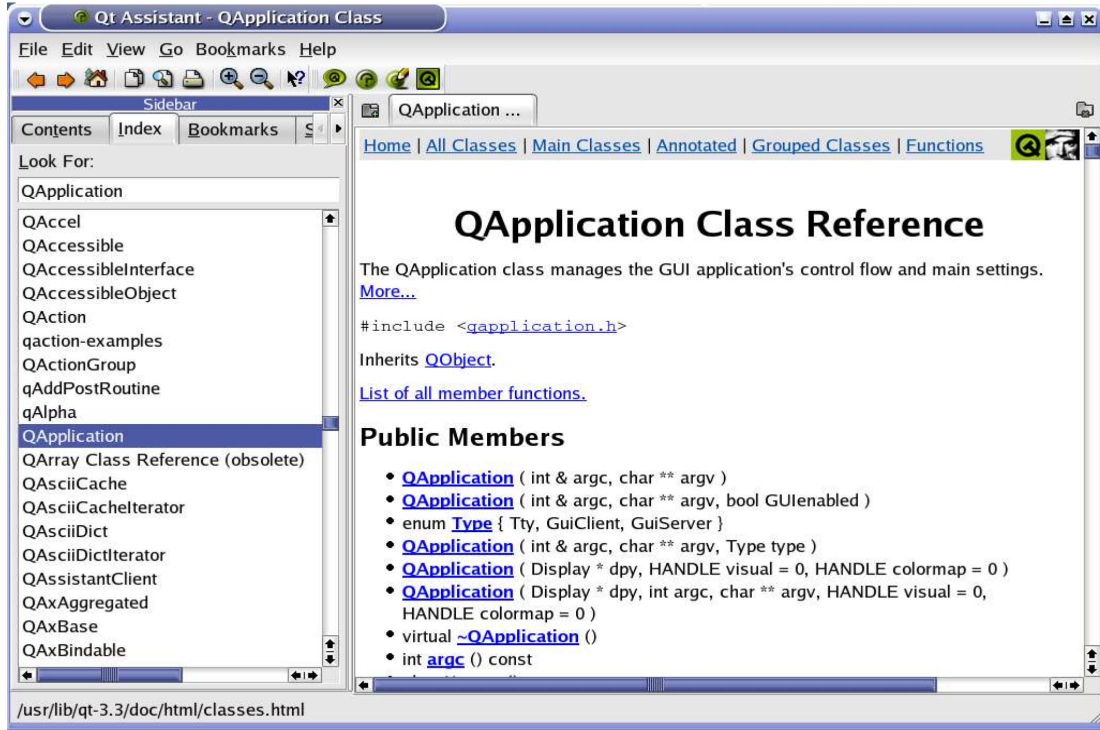
Windows XP



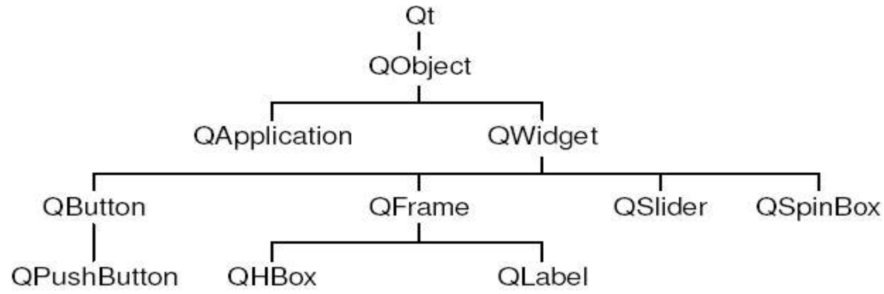
Mac

Şekil 1.8: SadeceWindpws XP veya Mac sistemıne mahsus tipler.

Diğer işletim sistemlerinin zıddına, Windows XP ve Mac görünümü ancak kendi işletim sistemleri altında verilebilir çünkü Qt bu görünümleri verirken sistemin kendisine dayanmaktadır.



Şekil 1.9: Qt assisstant



Şekil 1.9: Şu ana kadar karşılaştığımız Qt sınıflarının soy ağacı